

CLAIMS

1. (CURRENTLY AMENDED) A memory for storing information related to a B tree structure, wherein the B tree structure is used to access data records from a database system, wherein a set of the data records have duplicate keys, said memory comprising:
a plurality of interconnected nodes having a root node, index nodes and leaf nodes;

wherein a leaf node is configured to store a first key corresponding to first data in a first data page;

wherein data pages store a second key and a third key separately from the root node, index nodes and leaf nodes;

wherein the second key and the third key that are stored on the data pages are duplicate keys of the first key that is stored in the leaf node;

wherein the first key points to the second key;

wherein the second key points to the third key;

whereby the first, second and third keys are used by a data searching system for searching the set of the data records.

2. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the data pages include the first data page having the second key and a second data page having the third key, wherein said first data page and second data page comprise the same page.

3. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the data pages include the first data page having the second key and a second data page having the third key, wherein said first data page and second data page comprise different pages.

4. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the data pages include the first data page having the second key and a second data page having the third key,

wherein the second data page includes second data,

wherein said first data and second data are the same.

5. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the data pages include the first data page having the second key and a second data page having the third key,

wherein the second data page includes second data,

wherein said first data and second data are different.

6. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the data pages include the first data page having the second key and a second data page having the third key,

wherein the second data page includes second data,

wherein said first data has variable length.

7. (PREVIOUSLY PRESENTED) The memory of claim 6 wherein said second data has variable length.

8. (PREVIOUSLY PRESENTED) The memory of claim 7 wherein degree of the leaf nodes is not substantially affected by the variable length of the first and second data.

9. (PREVIOUSLY PRESENTED) The memory of claim 8 wherein degree of the leaf nodes is not substantially affected because the first and second data are stored separate from the leaf nodes.

10. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein said plurality of leaf nodes are maintained in sequential order and with a doubly linked list which connects each of said leaf node with its sibling nodes.

11. (PREVIOUSLY PRESENTED) The memory of claim 10 wherein the B-tree is configured to operate with a find operation.

12. (PREVIOUSLY PRESENTED) The memory of claim 10 wherein the B-tree is configured to operate with a find-next operation.

13. (PREVIOUSLY PRESENTED) The memory of claim 10 wherein the B-tree is configured to operate with a find-previous operation.

14. (PREVIOUSLY PRESENTED) The memory of claim 10 wherein the B-tree is configured to operate with a find-first operation.

15. (PREVIOUSLY PRESENTED) The memory of claim 10 wherein the B-tree is configured to operate with a find-last operation.
16. (PREVIOUSLY PRESENTED) The memory of claim 10 wherein the B-tree is configured to operate with an insert operation.
17. (PREVIOUSLY PRESENTED) The memory of claim 10 wherein the B-tree is configured to operate with a delete operation.
18. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein data associated with the first and second keys are stored separate from the leaf nodes.
19. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the first and second keys each have a corresponding unique data record value.
20. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein substantially concurrently executing processes update the first and second keys at approximately the same time without being locked out by another process because their associated data is stored on different data pages.
21. (PREVIOUSLY PRESENTED) The memory of claim 20 wherein the processes are threads.

22. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the data pages include the first data page having the second key and a second data page having the third key,

wherein the second data page includes second data,

wherein page and offset for the second key's value follow the second data on the second data page.

23. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein each page has associated with it a lock handle, wherein because the B-tree is self-balancing, an insert operation to the B-tree avoids locking the entire B-tree or subtree.

24. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the leaf nodes contain more than two key-value entries.

25. (PREVIOUSLY PRESENTED) The memory of claim 1 wherein the data pages include the first data page having the second key and a second data page having the third key,

wherein the second data page includes second data,

wherein the second key points to third data that is stored on a third data page.

26. (PREVIOUSLY PRESENTED) The memory of claim 25 wherein the third data is stored on the second data page.

27. (CURRENTLY AMENDED) A computer-implemented method for concurrent execution of a plurality of transactions in a database system containing a plurality of data records, wherein a set of the data records have duplicate keys, said method comprising:

storing said plurality of data records in a B* tree structure with a plurality of index nodes and a plurality of leaf nodes;

wherein a leaf node is configured to store a first key corresponding to first data in a first data page;

wherein data pages store a second key and a third key separately from the root node, index nodes and leaf nodes;

wherein the second key and the third key that are stored on the data pages are duplicate keys of the first key that is stored in the leaf node;

wherein the first key points to the second key;

wherein the second key points to the third key;

implementing said plurality of transactions by concurrently locating and operating on the target data records stored in said data pages through use of said B* tree structure by a data searching system.

28. (ORIGINAL) The method of claim 27 wherein said step of implementing said plurality of transactions further includes implementing a concurrency control protocol.

29. (PREVIOUSLY PRESENTED) The method of claim 28 wherein the data pages include the first data page having the second key and a second data page having the third key,

wherein the second data page includes second data,

wherein the concurrency control protocol controls a first of said transactions to access the first data in the first data page and concurrently a second of said transactions to access the second data in the second data page, wherein said first data and second data have the same key.

30. (ORIGINAL) The method of claim 28 wherein the concurrency control protocol is a lock-based protocol.

31. (ORIGINAL) The method of claim 28 wherein the lock-based protocol releases locks on index nodes and leaf nodes when the data page is identified.

32. (CURRENTLY AMENDED) A computer-readable medium for concurrent execution of a plurality of transactions in a database system containing a plurality of data records, wherein a set of the data records have duplicate keys, comprising instructions for:

storing said plurality of data records within a B* tree structure that has a plurality of index nodes and a plurality of leaf nodes;

wherein a leaf node is configured to store a first key corresponding to first data in a first data page;

wherein data pages store a second key and a third key separately from the root node, index nodes and leaf nodes;

wherein the second key and the third key that are stored on the data pages are duplicate keys of the first key that is stored in the leaf node;

wherein the first key points to the second key;

wherein the second key points to the third key;

implementing said plurality of transactions by concurrently locating and operating on the target data records stored in said data pages through use of the first, second and third keys by a data searching system.

33. (CURRENTLY AMENDED) An information processing system in database application, comprising:

a data store to store a plurality of data records with a first set of data records having duplicate keys, said plurality of data records stored in a B* tree structure with a plurality of index nodes and a plurality of leaf nodes;

wherein a leaf node is configured to store a first key corresponding to first data in a first data page;

wherein data pages store a second key and a third key separately from the root node, index nodes and leaf nodes;

wherein the second key and the third key that are stored on the data pages are duplicate keys of the first key that is stored in the leaf node;

wherein the first key points to the second key;

wherein the second key points to the third key;

whereby an engine for implementing a plurality of transactions are implemented by concurrently locating and operating on the data records stored in the data pages through use of the B* tree structure by a data searching system;

a concurrency-control manager for implementing a concurrency control protocol through use of the B* tree structure.

34. (PREVIOUSLY PRESENTED) The system of claim 1, wherein the third key points to the second key;

wherein the second key points to the first key.